RV1103/RV1106 RKNN SDK 快速上手指南

文件标识：RK-JC-YF-415

发布版本：V2.4.0

日期：2026-01-17

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

本文档按"现状"提供，瑞芯微电子股份有限公司（"本公司"，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

"Rockchip"、"瑞芯微"、"瑞芯"均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2026 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：　福建省福州市铜盘路软件园A区18号

网址：　www.rock-chips.com

客户服务电话：　+86-4007-700-590

客户服务传真：　+86-591-83951833

客户服务邮箱：　fae@rock-chips.com

前言

概述

本文介绍RV1103B/RV1103/RV1106B/RV1106 RKNN SDK 快速上手指南。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本	修改人	修改日期	修改说明	核定人
V1.6.0	HPC	2023-11-28	初始版本	熊伟
V2.0.0-beta0	HPC	2024-03-15	更新版本号	熊伟
V2.1.0	HPC	2024-08-01	增加 RV1103B 支持	熊伟
V2.2.0	HPC	2024-09-04	增加 RV1106B 支持	熊伟
V2.3.0	HPC	2024-11-06	增加通过 pip 源安装 RKNN-Toolkit2 的支持	熊伟
V2.3.2	HPC	2025-04-03	更新版本号	熊伟
V2.4.0	HPC	2026-01-17	更新版本号	熊伟

目 录

RV1103/RV1106 RKNN SDK 快速上手指南
1 主要说明
2 准备开发板
  2.1 开发板和连接工具介绍
  2.2 连接开发板
3 准备开发环境
  3.1 下载 RKNN 相关仓库
  3.2 安装 RKNN-Toolkit2 环境
    3.2.1 安装 Python
      3.2.1.1 安装 Miniforge
      3.2.1.2 使用 Miniforge 创建 Python 环境
    3.2.2 安装 RKNN-Toolkit2
    3.2.3 验证是否安装成功
  3.3 安装编译工具
    3.3.1 安装 CMake
    3.3.2 安装编译器
      3.3.2.1 确认开发板系统架构
      3.3.2.2 安装 GCC 交叉编译器
  3.4 安装板端 RKNPU2 环境
    3.4.1 确认 RKNPU2 驱动版本
    3.4.2 检查 RKNPU2 环境是否安装
    3.4.3 安装/更新 RKNPU2 环境
4 运行示例程序
  4.1 RKNN Model Zoo 介绍
  4.2 RKNN Python Demo 使用方法
    4.2.1 准备模型
    4.2.2 模型转换
    4.2.3 运行 RKNN Python Demo
    4.2.4 数据集精度评估（可选）
  4.3 RKNN C Demo 使用方法
    4.3.1 准备模型
    4.3.2 模型转换
    4.3.3 运行 RKNN C Demo
      4.3.3.1 编译
      4.3.3.2 推送文件到板端
      4.3.3.3 板端运行 Demo

4.3.3.4 查看结果
5 Docker 中运行 RKNN Python Demo（可选）
5.1 安装 Docker
5.2 在 Docker 中安装 RKNN-Toolkit2 环境
5.2.1 准备 RKNN-Toolkit2 镜像
5.2.2 查询镜像信息
5.3 RKNN Python Demo 使用方法
6 常见问题
6.1 命令 adb devices 查看不到设备
6.2 手动启动 rknn_server 服务
6.3 内存不足导致模型加载失败
7 参考文档

1 主要说明

此文档面向零基础用户详细介绍如何快速在计算机上使用 RKNN-Toolkit2 完成模型转换，并通过 RKNPU2 部署到 Rockchip 开发板上。本文所用示例已集成到 RKNN Model Zoo 中。

支持的平台：

- RV1103
- RV1103B
- RV1106
- RV1106B

2 准备开发板

本章将介绍如何将开发板连接到计算机，分为两个部分：

- 开发板和连接工具介绍
- 连接开发板

2.1 开发板和连接工具介绍

1. 开发板

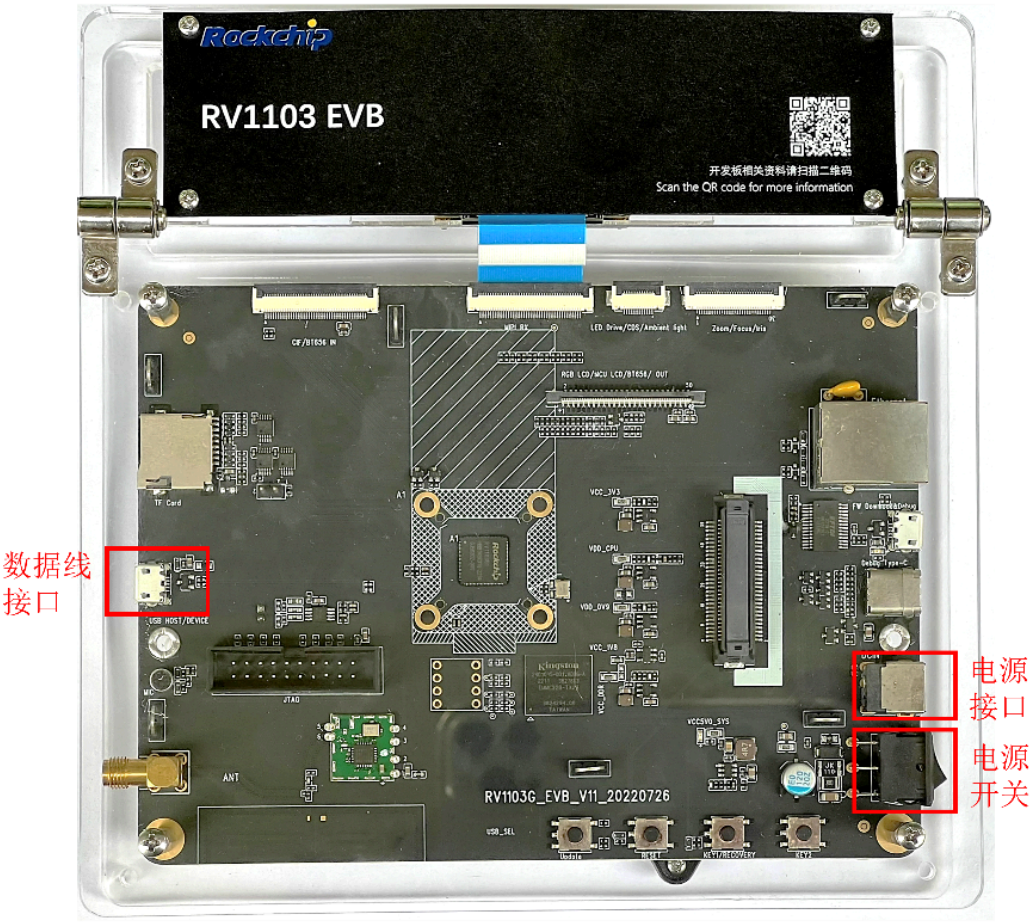

图2-1 RV1103开发板

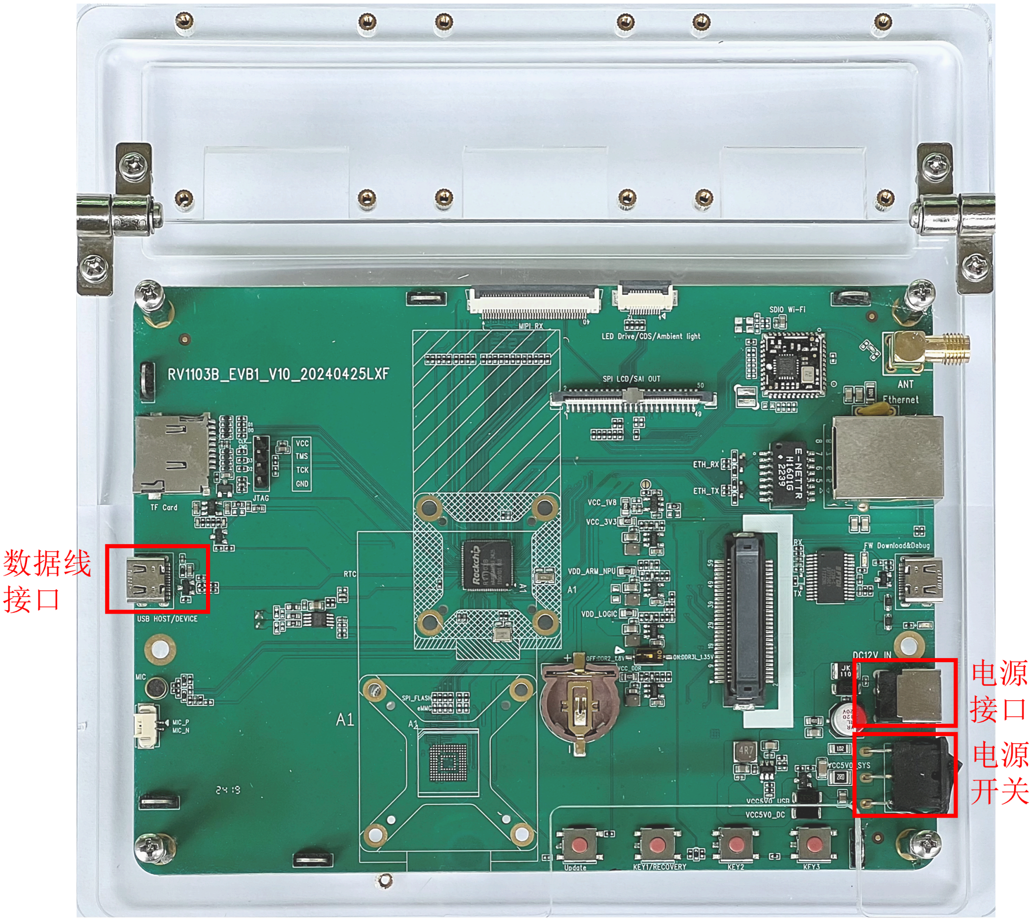

图2-2 RV1103B开发板

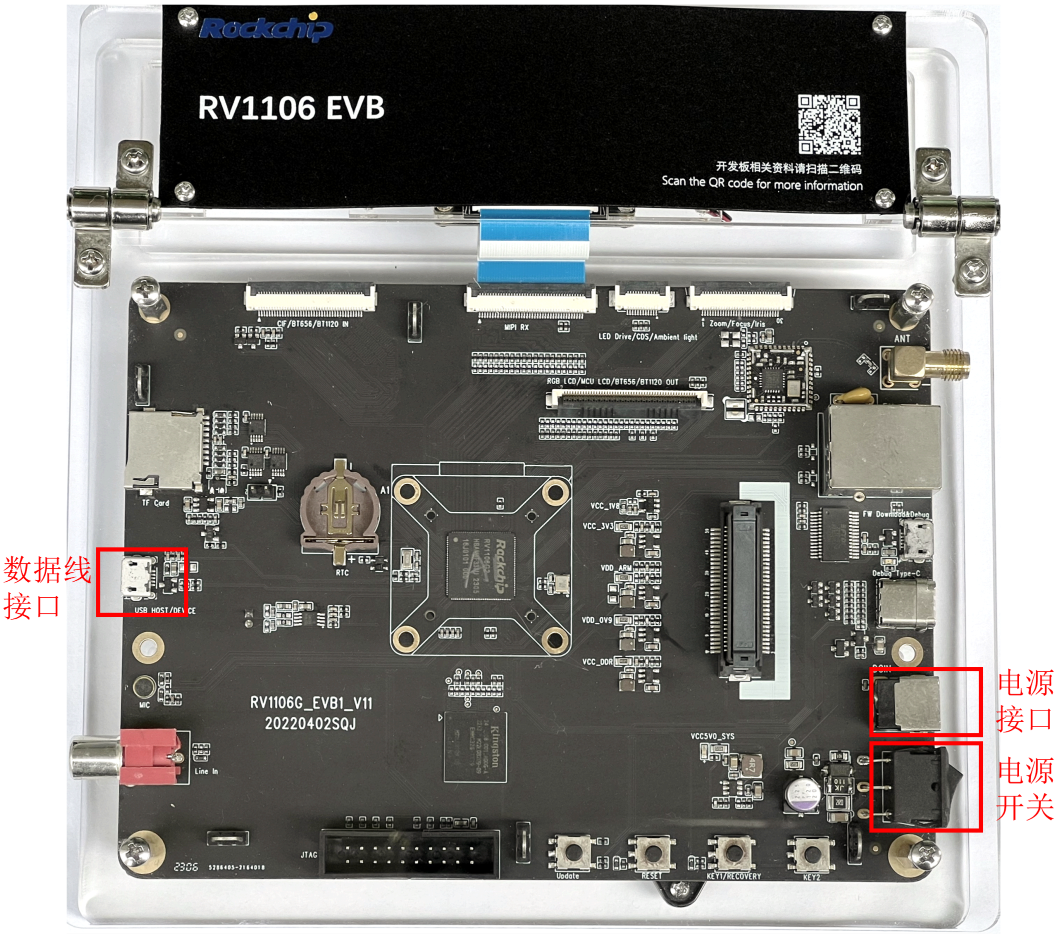

图2-3 RV1106开发板

2. 连接开发板和计算机的数据线

图2-4 Micro USB 数据线

3. 电源适配器

图2-5 输出 12V-2A 的电源适配器

2.2 连接开发板

下面以 RV1106 为例说明如何将开发板连至计算机：

1. 准备一台操作系统为 Ubuntu18.04 / Ubuntu20.04 / Ubuntu22.04 的计算机。
2. 找到下图中电源接口的位置，连接开发板电源适配器。
3. 使用数据线连接开发板与计算机（注意，由于生产批次不同，开发板的数据线接口类型和位置可能会发生变化。通常情况下，开发板上印有 OTG 字样的接口为数据线接口）。
4. 打开电源开关，等待开发板系统启动完成。

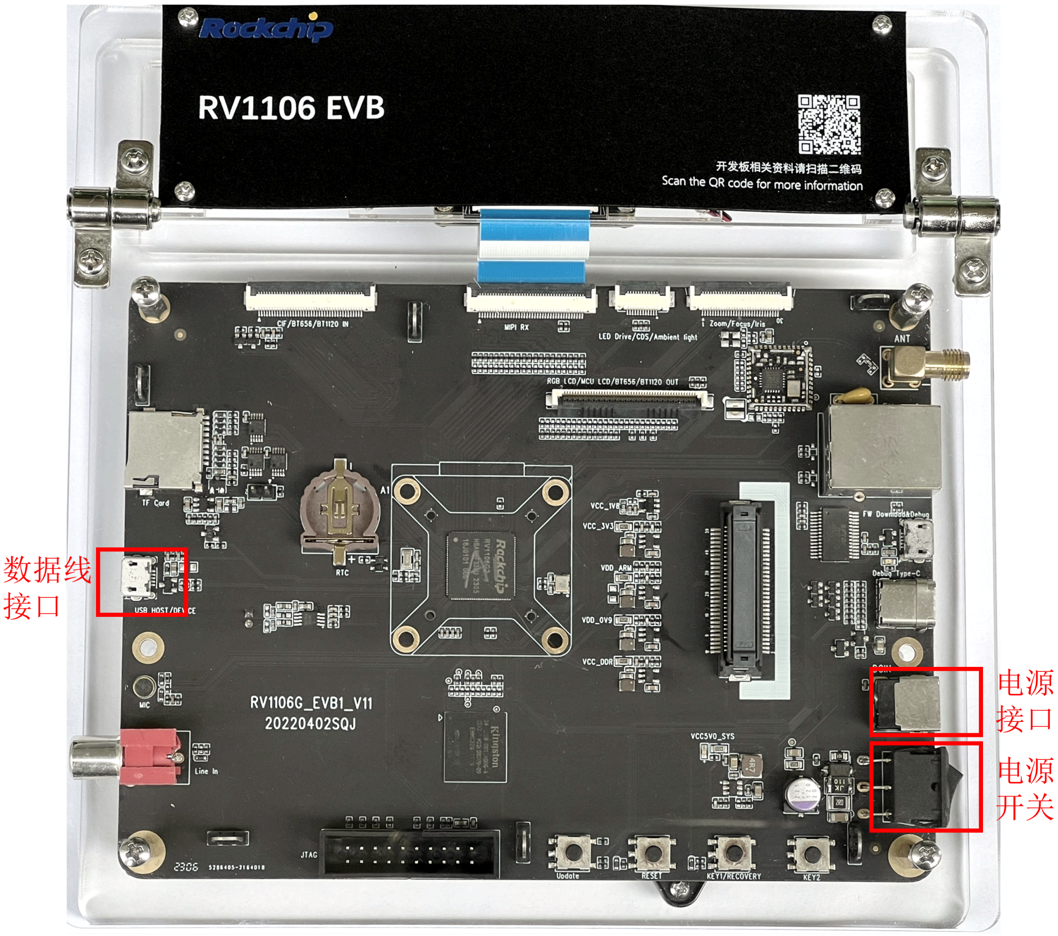

图2-6 RV1106开发板

5. 查看开发板是否连接至计算机

   在计算机的终端窗口（命令行界面）中，执行以下命令：

```
# 如果没有安装过 adb，请先使用 sudo apt install adb 安装
adb devices
```

连接成功时输出信息如下，其中 421ca310c9b9242b 为 RV1106 的设备 ID。若无设备显示请参考第 6.1 章节进行排查。

```
$ adb devices
List of devices attached
421ca310c9b9242b    device
```

3 准备开发环境

本章介绍如何在计算机中直接安装开发环境，后续的示例程序运行过程也将以直接安装为例说明。如果需要在 Docker 环境中运行示例程序，可以参考第5章中的内容准备开发环境。

本章分为四个部分：

- 下载 RKNN 相关仓库
- 安装RKNN-Toolkit2 环境

- 安装编译工具
- 安装板端 RKNPU2 环境

3.1 下载 RKNN 相关仓库

建议新建一个目录用来存放 RKNN 仓库，例如新建一个名称为 Projects 的文件夹，并将 RKNN-Toolkit2 和 RKNN Model Zoo 仓库存放至该目录下，参考命令如下：

```
# 新建 Projects 文件夹
mkdir Projects

# 进入该目录
cd Projects

# 下载 RKNN-Toolkit2 仓库
git clone https://github.com/airockchip/rknn-toolkit2.git --depth 1
# 下载 RKNN Model Zoo 仓库
git clone https://github.com/airockchip/rknn_model_zoo.git --depth 1
# 注意：
# 1.参数 --depth 1 表示只克隆最近一次 commit
# 2.如果遇到 git clone 失败的情况，也可以直接在 github 中下载压缩包到本地，然后解压至该目录
```

整体目录结构如下：

```
Projects
├── rknn-toolkit2
│   ├── doc
│   ├── rknn-toolkit2
│   │   ├── packages
│   │   ├── docker
│   │   └── ...
│   ├── rknpu2
│   │   ├── runtime
│   │   └── ...
│   └── ...
└── rknn_model_zoo
    ├── datasets
    ├── examples
    └── ...
```

3.2 安装 RKNN-Toolkit2 环境

3.2.1 安装 Python

如果系统中没有安装 Python 3.8（建议版本），或者同时有多个版本的 Python 环境，建议使用 Miniforge 创建新的 Python 3.8 环境。

3.2.1.1 安装 Miniforge

在计算机的终端窗口中执行以下命令，检查是否安装 Miniforge，若已安装则可省略此节步骤。

```
conda -V
# 参考输出信息: conda 23.3.1 , 表示 Miniforge conda 版本为 23.3.1
# 如果提示 conda: command not found, 则表示未安装 Miniforge
```

如果没有安装 Miniforge，可以通过下面的链接下载 Miniforge 安装包：

```
wget -c https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-x86_64.sh
```

然后通过以下命令安装 Miniforge

```
chmod 777 Miniforge3-Linux-x86_64.sh
bash Miniforge3-Linux-x86_64.sh
```

3.2.1.2 使用 Miniforge 创建 Python 环境

在计算机的终端窗口中，执行以下命令进入 Miniforge base 环境：

```
source ~/miniforge3/bin/activate # Miniforge 安装的目录
# 成功后, 命令行提示符会变成以下形式:
# (base) xxx@xxx:~$
```

通过以下命令创建名称为 toolkit2 的 Python 3.8 环境：

```
conda create -n toolkit2 python=3.8
```

激活 toolkit2 环境，后续将在此环境中安装 RKNN-Toolkit2：

```
conda activate toolkit2
# 成功后, 命令行提示符会变成以下形式:
# (toolkit2) xxx@xxx:~$
```

3.2.2 安装 RKNN-Toolkit2

激活 toolkit2 环境后，可通过 pip 源或者本地 wheel 包这两种方式安装 RKNN-Toolkit2：

- 通过 pip 源安装

```
pip install rknn-toolkit2 -i https://pypi.org/simple

# 如果已安装 RKNN-Toolkit2, 可通过以下命令升级 RKNN-Toolkit2
pip install rknn-toolkit2 -i https://pypi.org/simple --upgrade
```

- 通过本地 wheel 包安装

```
# 进入 rknn-toolkit2 目录
cd Projects/rknn-toolkit2/rknn-toolkit2

# 请根据不同的 python 版本及处理器架构，选择不同的 requirements 文件：
# 其中 cpxx 是 python 版本号
pip install -r packages/x86_64/requirements_cpxx.txt
# pip install -r packages/arm64/arm64_requirements_cpxx.txt

# 安装 RKNN-Toolkit2
# 请根据不同的 python 版本及处理器架构，选择不同的 wheel 安装包文件：
# 其中 x.x.x 是 RKNN-Toolkit2 版本号，cpxx 是 python 版本号
pip install packages/x86_64/rknn_toolkit2-x.x.x-cpxx-cpxx-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
# pip install packages/arm64/rknn_toolkit2-x.x.x-cpxx-cpxx-manylinux_2_17_aarch64.manylinux2014_aarch64.whl
```

3.2.3 验证是否安装成功

执行以下命令，若没有报错，则代表 RKNN-Toolkit2 环境安装成功。

```
# 进入 Python 交互模式
python

# 导入 RKNN 类
from rknn.api import RKNN
```

如果安装失败，请查阅 《Rockchip_RKNPU_User_Guide_RKNN_SDK_CN.pdf》文档中的第 10.2 章节 “工具安装问题”，其中详细介绍了 RKNN-Toolkit2 环境安装失败的解决方法。

3.3 安装编译工具

3.3.1 安装 CMake

在计算机的终端中，执行以下命令：

```
# 更新包列表
sudo apt update

# 安装 cmake
sudo apt install cmake
```

3.3.2 安装编译器

3.3.2.1 确认开发板系统架构

可以在计算机端执行以下命令查询系统架构：

```
adb shell uname -a
```

该命令的参考输出信息如下，其中 armv7l 表示基于 ARMv7 架构，属于armhf 架构（架构类型在后续编译 RKNN C Demo 时会用到）

```
$ adb shell uname -a
Linux Rockchip 5.10.160 #1 Tue Oct 24 18:52:11 CST 2023 armv7l GNU/Linux
```

3.3.2.2 安装 GCC 交叉编译器

- GCC 下载地址：https://console.zbox.filez.com/l/H1fV9a (提取码是：rknn)
- 解压软件包

  建议将 GCC 软件包解压到 Projects 的文件夹中。存放位置如下：

```
Projects
├── rknn-toolkit2
├── rknn_model_zoo
└── arm-rockchip830-linux-uclibcgnueabihf # 此路径在后面编译RKNN C Demo时会用到
```

此时，GCC 编译器的路径是 Projects/arm-rockchip830-linux-uclibcgnueabihf/bin/arm-rockchip830-linux-uclibcgnueabihf

3.4 安装板端 RKNPU2 环境

为了方便描述，后续文档使用板端来表示开发板端。

3.4.1 确认 RKNPU2 驱动版本

可以在板端执行以下命令查询 RKNPU2 驱动版本：

```
dmesg | grep -i rknpu
```

如下图所示，当前 RKNPU2 驱动版本为 0.8.5。

```
# dmesg | grep -i rknpu
[    4.431371] RKNPU ff660000.npu: RKNPU: rknpu iommu device-tree entry not found!, using non-iommu mode
[    4.432210] RKNPU ff660000.npu: RKNPU: Initialized RKNPU driver: v0.8.5 for 20230202
[    4.432314] RKNPU ff660000.npu: dev_pm_opp_set_regulators: no regulator (rknpu) found: -19
```

图3-1 RKNPU2 驱动版本信息

Rockchip 开发板的官方固件均自带 RKNPU2 驱动。若以上命令查询不到 NPU 驱动版本，则可能使用的是第三方固件，其中可能没有安装NPU驱动。如果有固件源码，可以在 kernel config 中将 CONFIG_ROCKCHIP_RKNPU 选项的值改成 y 以集成 NPU 驱动，然后重新编译内核驱动并烧录。建议 RKNPU2 驱动版本 >= 0.9.2。

3.4.2 检查 RKNPU2 环境是否安装

RKNN-Toolkit2 的连板调试功能要求板端已安装 RKNPU2 环境，并且启动 rknn_server 服务。以下是 RKNPU2 环境中的两个基本概念：

- RKNN Server：一个运行在开发板上的后台代理服务。该服务的主要功能是调用板端 Runtime 对应的接口处理计算机通过USB传输过来的数据，并将处理结果返回给计算机。

- RKNPU2 Runtime 库（librknnmrt.so）：主要职责是负责在系统中加载 RKNN 模型，并通过调用专用的神经处理单元（NPU）执行 RKNN 模型的推理操作。

如果板端没有安装 RKNN Server 和 Runtime 库，或者 RKNN Server 和 Runtime 库的版本不一致，都需要重新安装 RKNPU2 环境。（注意：1. 若使用动态维度输入的 RKNN 模型，则要求 RKNN Server 和 Runtime 库版本 >= 1.5.0。2. 要保证 RKNN Server 、Runtime 库的版本、RKNN-Toolkit2 的版本是一致的，建议都安装最新的版本）

通常情况下，开发板默认已经安装版本一致的 RKNPU2 环境，可以通过下面命令确认（如果没有安装 RKNPU2 环境或者版本不一致，请按照下一节中的步骤来安装/更新 RKNPU2 环境）：

1. 检查 RKNPU2 环境是否安装

如果能够启动 rknn_server 服务，则代表板端已经安装 RKNPU2 环境。

```
# 进入板端
adb shell

# 启动 rknn_server
restart_rknn.sh
```

如果出现以下输出信息，则代表启动rknn_server服务成功，即已经安装RKNPU2环境。

```
start rknn server, version: x.x.x
```

2. 检查版本是否一致

```
# 查询rknn_server版本
strings /oem/usr/bin/rknn_server | grep -i "rknn_server version"

# 查询librknnmrt.so库版本
strings /oem/usr/lib/librknnmrt.so | grep -i "librknnmrt version"
```

如果出现以下输出信息，则代表rknn_server版本为 x.x.x，librknnmrt.so 的版本为 x.x.x。

```
rknn_server version: x.x.x
librknnmrt version: x.x.x
```

3.4.3 安装/更新 RKNPU2 环境

注：如果已经安装版本一致 RKNPU2 环境，则此节内容可以跳过。

进入 rknpu2 目录，使用 adb 工具将相应的 rknn_server 和 librknnmrt.so 推送至板端，然后启动 rknn_server，参考命令如下：

```
# 进入 rknpu2 目录
cd Projects/rknn-toolkit2/rknpu2

# 推送 rknn_server 到板端
adb push runtime/Linux/rknn_server/armhf-uclibc/usr/bin/* /oem/usr/bin

# 推送 librknnmrt.so
```

```
adb push runtime/Linux/librknn_api/armhf-uclibc/librknnmrt.so /oem/usr/lib

# 进入板端
adb shell

# 赋予可执行权限
chmod +x /oem/usr/bin/rknn_server
chmod +x /oem/usr/bin/start_rknn.sh
chmod +x /oem/usr/bin/restart_rknn.sh

# 重启 rknn_server 服务
restart_rknn.sh
```

4 运行示例程序

本章将介绍如何快速在开发板上运行示例程序，内容分为三个部分：

- RKNN Model Zoo介绍
- RKNN Python Demo 使用方法
- RKNN C Demo 使用方法

4.1 RKNN Model Zoo 介绍

RKNN Model Zoo 提供了示例代码，旨在帮助用户快速在 Rockchip 的开发板上运行各种常用模型，整个工程的目录结构如下：

```
rknn_model_zoo
├── 3rdparty # 第三方库
├── datasets # 数据集
├── examples # 示例代码
├── utils # 常用方法，如文件操作，画图等
├── build-android.sh # 用于目标为 Android 系统开发板的编译脚本
├── build-linux.sh # 用于目标为 Linux 系统开发板的编译脚本
└── ...
```

其中，examples 目录包括了一些常用模型的示例，例如 MobileNet 和 YOLO 等。每个模型示例提供了 Python 和 C/C++ 两个版本的示例代码（为了方便描述，后续用 RKNN Python Demo 和 RKNN C Demo 来表示）。以 YOLOv5 模型为例，其目录结构如下：

```
rknn_model_zoo
├── examples
│   └── yolov5
│       ├── cpp # C/C++ 版本的示例代码
│       ├── model # 模型、测试图片等文件
│       ├── python # 模型转换脚本和Python版本的示例代码
│       └── README.md
└── ...
```

4.2 RKNN Python Demo 使用方法

下面以 RV1106 平台为例，介绍 YOLOv5 RKNN Python Demo 的使用方法。

注：不同的 RKNN Python Demo 用法存在差异，请按照各自目录下 README.md 中的步骤运行。

4.2.1 准备模型

进入 rknn_model_zoo/examples/yolov5/model 目录，运行 download_model.sh 脚本，该脚本将下载一个可用的 YOLOv5 ONNX 模型，并存放在当前 model 目录下，参考命令如下：

```
# 进入 rknn_model_zoo/examples/yolov5/model 目录
cd Projects/rknn_model_zoo/examples/yolov5/model

# 运行 download_model.sh 脚本，下载 yolov5 onnx 模型
# 例如，下载好的 onnx 模型存放路径为 model/yolov5s_relu.onnx
./download_model.sh
```

4.2.2 模型转换

进入 rknn_model_zoo/examples/yolov5/python 目录，运行 convert.py 脚本，该脚本将原始的 ONNX 模型转成 RKNN 模型，参考命令如下：

```
# 进入 rknn_model_zoo/examples/yolov5/python 目录
cd Projects/rknn_model_zoo/examples/yolov5/python

# 运行 convert.py 脚本，将原始的 ONNX 模型转成 RKNN 模型
# 用法: python convert.py model_path [rv1103|rv1103b|rv1106|rv1106b] [i8/fp] [output_path]
python convert.py ../model/yolov5s_relu.onnx rv1106 i8 ../model/yolov5s_relu.rknn
# 注：rv1103、rv1106和rv1103b、rv1106b生成的模型不能共用
```

4.2.3 运行 RKNN Python Demo

进入 rknn_model_zoo/examples/yolov5/python 目录，运行 yolov5.py 脚本，便可通过连板调试的方式在板端运行 YOLOv5 模型，参考命令如下：

```
# 进入 rknn_model_zoo/examples/yolov5/python 目录
cd Projects/rknn_model_zoo/examples/yolov5/python

# 运行 yolov5.py 脚本，在板端运行 yolov5 模型
# 用法: python yolov5.py --model_path {rknn_model} --target {target_platform} --img_show
# 其中，如果带上 --img_show 参数，则会显示结果图片
# 注：这里以 RV1106 平台为例，如果是其他开发板，则需要修改命令中的平台类型
python yolov5.py --model_path ../model/yolov5s_relu.rknn --target rv1106 --img_show

# 如果想先在计算机端运行原始的 onnx 模型，可以参考以下命令
# 用法: python yolov5.py --model_path {onnx_model} --img_show
python yolov5.py --model_path ../model/yolov5s_relu.onnx --img_show
```

默认输入图片是 model/bus.jpg，输出图片如下所示：

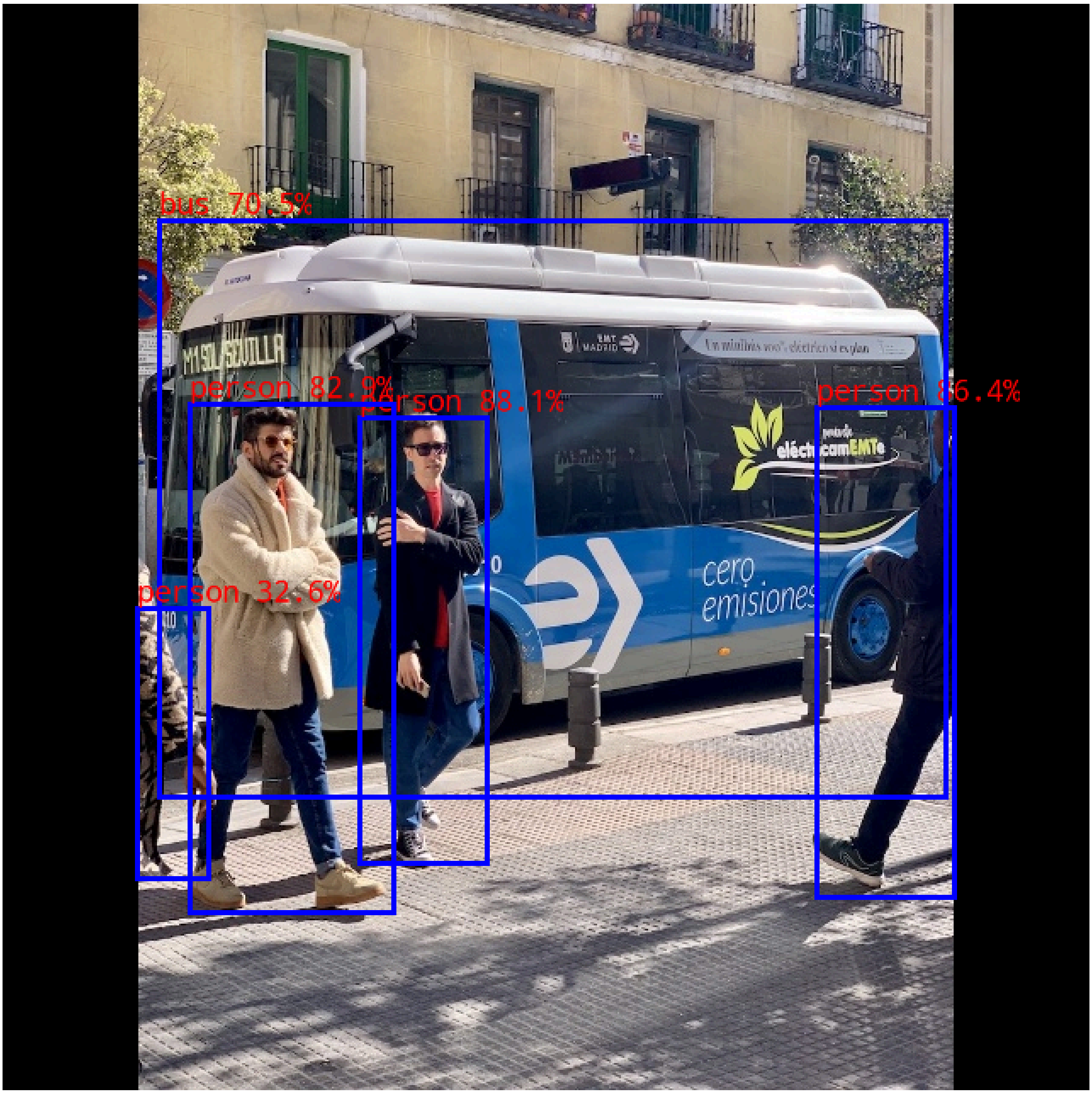

图4-1 RKNN Python Demo 输出图片

4.2.4 数据集精度评估（可选）

rknn_model_zoo/datasets 目录存放数据集，用于精度评估，需要先下载评估数据集并保存至该目录。例如，对于YOLOv5模型，需要下载 COCO 数据集。进入 rknn_model_zoo/datasets/COCO 目录，运行 download_eval_dataset.py 脚本，该脚本将下载 val2017 数据集，并存放在当前 COCO 目录下，参考命令如下：

```
# 进入 rknn_model_zoo/datasets/COCO 目录
cd Projects/rknn_model_zoo/datasets/COCO

# 运行 download_eval_dataset.py 脚本，下载 COCO 数据集
python download_eval_dataset.py
```

进行数据集精度评估时，需要指定 --coco_map_test 参数，并指定评估数据集路径 --img_folder，参考命令如下：

```
# 请先安装 pycocotools
pip install pycocotools

# 进入 rknn_model_zoo/examples/yolov5/python 目录
cd Projects/rknn_model_zoo/examples/yolov5/python

# 运行 yolov5.py 脚本
python yolov5.py \
    --model_path ../model/yolov5s_relu.rknn \
    --target rv1106 \
    --img_folder ../../../datasets/COCO/val2017 \
    --coco_map_test
```

4.3 RKNN C Demo 使用方法

下面以 RV1106 平台为例，介绍 YOLOv5 RKNN C Demo 的使用方法。

注：不同的 RKNN C Demo 用法存在差异，请按照各自目录下 README.md 中的步骤运行。

4.3.1 准备模型

进入 rknn_model_zoo/examples/yolov5/model 目录，运行 download_model.sh 脚本，该脚本将下载一个可用的 YOLOv5 ONNX 模型，并存放在当前 model 目录下，参考命令如下：

```
# 进入 rknn_model_zoo/examples/yolov5/model 目录
cd Projects/rknn_model_zoo/examples/yolov5/model

# 运行 download_model.sh 脚本，下载 yolov5 onnx 模型
# 例如，下载好的 onnx 模型存放路径为 model/yolov5s_relu.onnx
./download_model.sh
```

4.3.2 模型转换

进入 rknn_model_zoo/examples/yolov5/python 目录，运行 convert.py 脚本，该脚本将原始的 ONNX 模型转成 RKNN 模型，参考命令如下：

```
# 进入 rknn_model_zoo/examples/yolov5/python 目录
cd Projects/rknn_model_zoo/examples/yolov5/python

# 运行 convert.py 脚本，将原始的 ONNX 模型转成 RKNN 模型
# 用法: python convert.py model_path [rv1103|rv1103b|rv1106|rv1106b] [i8/fp] [output_path]
python convert.py ../model/yolov5s_relu.onnx rv1106 i8 ../model/yolov5s_relu.rknn
# 注: rv1103、rv1106和rv1103b、rv1106b生成的模型不能共用
```

4.3.3 运行 RKNN C Demo

完整运行一个 RKNN C Demo，需要先将 C/C++ 源代码编译成可执行文件，然后将可执行文件、模型文件、测试图片等相关文件推送到板端上，最后在板端运行可执行文件。

4.3.3.1 编译

以 Linux 系统（armhf 架构） 的 RV1106 平台为例 （注：RV1103/RV1103B 平台也是 Linux 系统的开发板），需要使用 rknn_model_zoo 目录下的 build-linux.sh 脚本进行编译。在运行 build-linux.sh 脚本之前，需要指定编译器的路径 GCC_COMPILER 为本地的 GCC 编译器路径。即在 build-linux.sh 脚本中，需要加入以下命令：

```
# 添加到 build-linux.sh 脚本的开头位置即可
GCC_COMPILER=Projects/arm-rockchip830-linux-uclibcgnueabihf/bin/arm-rockchip830-linux-uclibcgnueabihf
```

然后在 rknn_model_zoo 目录下，运行 build-linux.sh 脚本，参考命令如下：

```
# 进入 rknn_model_zoo 目录
cd Projects/rknn_model_zoo

# 运行 build-linux.sh 脚本
# 用法:./build-linux.sh -t <target> -a <arch> -d <build_demo_name> [-b <build_type>] [-m]
# -t : target (rv1103/rv1106) # 平台类型, rv1103b、rv1106b和rv1103、rv1106除模型外其他代码可共用
# -a : arch (aarch64/armhf) # 板端系统架构
# -d : demo name # 对应 examples 目录下子文件夹的名称，如yolov5、mobilenet
# -b : build_type(Debug/Release)
# -m : enable address sanitizer, build_type need set to Debug
./build-linux.sh -t rv1106 -a armhf -d yolov5
```

4.3.3.2 推送文件到板端

编译完成后，会在 rknn_model_zoo 目录下产生 install 文件夹， 其中有编译好的可执行文件，以及测试图片等相关文件。参考目录结构如下：

```
install
├── rv1106_linux_armhf # rv1106 平台
    └── rknn_yolov5_demo
        ├── lib # 依赖库
        ├── model # 存放模型、测试图片等文件
        └── rknn_yolov5_demo # 可执行文件
```

执行以下命令，将文件推送到板端：

```
# 进入 rknn_model_zoo 目录
cd Projects/rknn_model_zoo

# 推送整个 rknn_yolov5_demo 文件夹到板端
# 注：rknn_yolov5_demo 文件夹下有一个同名的可执行文件 rknn_yolov5_demo
# 注：使用不同的模型和平台时，建议直接在 install 下找对应的路径
adb push install/rv1106_linux_armhf/rknn_yolov5_demo /data/
```

4.3.3.3 板端运行 Demo

执行以下命令，在板端运行可执行文件：

```
# 进入板端
adb shell

# 进入 rknn_yolov5_demo 目录
cd /data/rknn_yolov5_demo/

# 设置依赖库环境
export LD_LIBRARY_PATH=/data/rknn_yolov5_demo/lib

# 运行可执行文件
# 用法: ./rknn_yolov5_demo <model_path> <input_path>
./rknn_yolov5_demo model/yolov5s_relu.rknn model/bus.jpg
```

4.3.3.4 查看结果

默认情况下，输出图片保存路径为 rknn_yolov5_demo/out.png，可以通过 adb 工具从板端拉取到本地查看，在本地计算机的终端中，执行以下命令：

```
# 拉取到本地当前目录
adb pull /data/rknn_yolov5_demo/out.png .
```

输出图片如下所示：

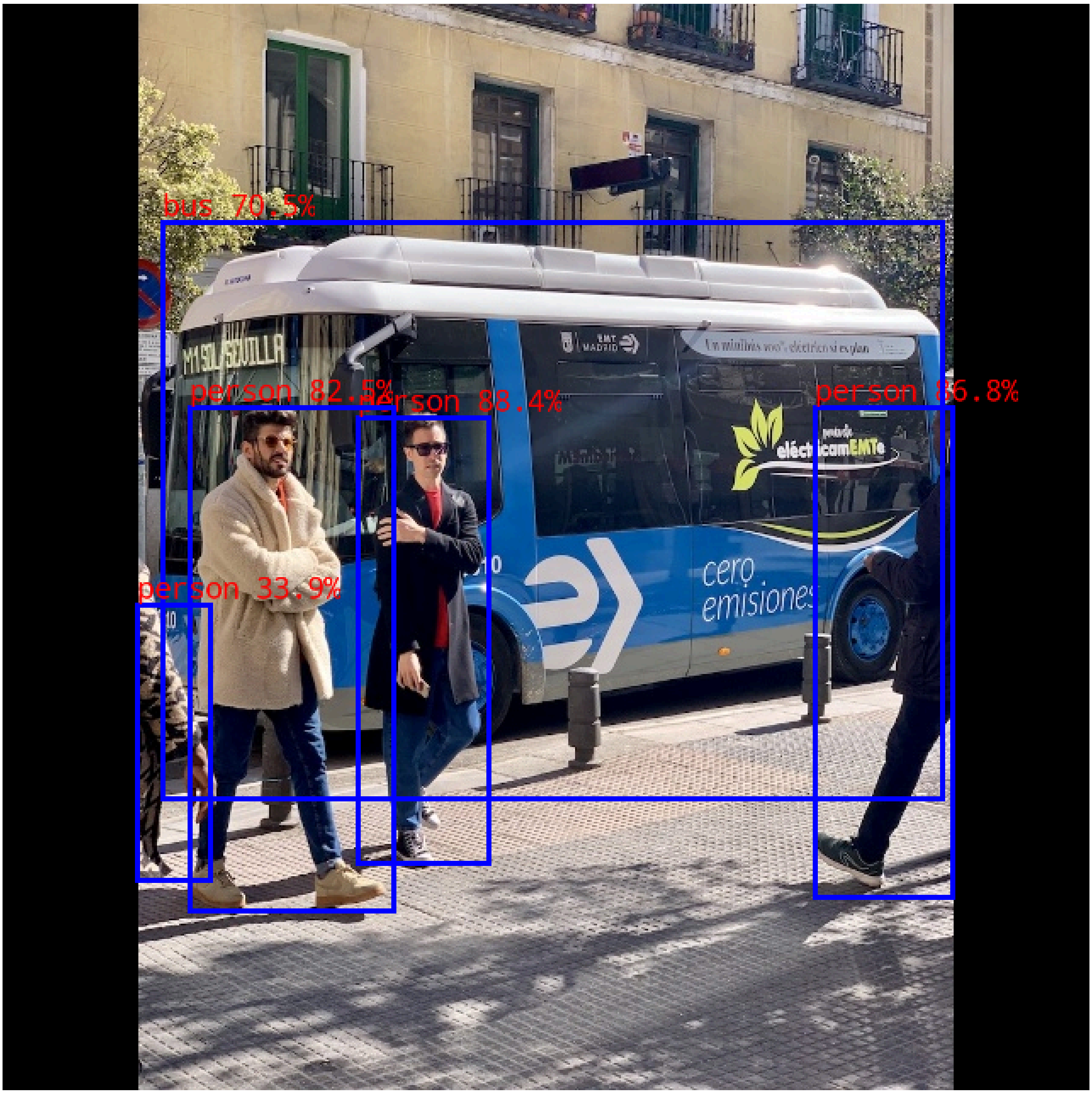

图4-2 RKNN C Demo 输出图片

5 Docker 中运行 RKNN Python Demo（可选）

如果需要在 Docker 环境中运行 RKNN Python Demo，可以参考本章中的内容准备开发环境。

请特别注意，这里提供了一个包含 RKNN-Toolkit2 环境的 Docker 镜像，允许用户在其中直接运行 RKNN Python Demo 而无需担心环境安装问题。但是，该 Docker 镜像中只包含纯净的 RKNN-Toolkit2 环境，仅适用于运行 RKNN Python Demo。

此章内容分为三个部分：

- 安装 Docker
- 在 Docker 中安装 RKNN-Toolkit2 环境
- RKNN Python Demo 使用方法

5.1 安装 Docker

如果已安装 Docker，可跳过此步骤。如果没有安装，请根据官方手册进行安装。

Docker 安装官方手册链接：https://docs.docker.com/install/linux/docker-ce/ubuntu/

注意事项：需要将用户添加到 docker 用户组。

```
# 创建docker用户组
sudo groupadd docker
# 把当前用户加入docker用户组
sudo usermod -aG docker $USER
# 更新激活docker用户组
newgrp docker

# 验证不需要sudo执行docker命令
docker run hello-world
```

成功安装的参考输出信息如下：

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:88ec0acaa3ec199d3b7eaf73588f4518c25f9d34f58ce9a0df68429c5af48e8d
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
```

5.2 在 Docker 中安装 RKNN-Toolkit2 环境

5.2.1 准备 RKNN-Toolkit2 镜像

本节介绍两种创建 RKNN-Toolkit2 镜像环境的方式，可任选一种方式进行创建。

1. 通过 Dockerfile 创建 RKNN Toolkit2 镜像

   在 RKNN-Toolkit2 工程中 docker/docker_file 文件夹下，提供了构建 RKNN-Toolkit2 开发环境的 Dockerfile 文件，用户通过 docker build命令创建镜像，如下所示：

```
# 注：以下 xx 和 x.x.x 代表版本号，请根据实际数值进行替换
cd Projects/rknn-toolkit2/rknn-toolkit2/docker/docker_file/ubuntu_xx_xx_cpxx

docker build -f Dockerfile_ubuntu_xx_xx_for_cpxx -t rknn-toolkit2:x.x.x-cpxx .
```

2. 通过加载已打包的 Docker 镜像文件创建 RKNN Toolkit2 镜像

   通过如下链接下载对应版本的 RKNN-Toolkit2 工程文件，解压后在 docker/docker_image 文件夹下提供了已打包所有开发环境的 Docker 镜像。

   Docker 镜像文件网盘下载链接：https://console.zbox.filez.com/l/I00fc3 （提取码：rknn）

   执行以下命令加载对应 Python 版本的镜像文件。

```
# 注：x.x.x 代表 RKNN-Toolkit2 的版本号，cpxx 代表的是 Python 的版本
docker load --input rknn-toolkit2-x.x.x-cpxx-docker.tar.gz
```

5.2.2 查询镜像信息

创建或加载镜像成功后，可以通过以下命令查看 docker 的镜像信息。

```
docker images
```

相应的 RKNN-Toolkit2 镜像信息显示如下：

```
REPOSITORY          TAG            IMAGE ID         CREATED         SIZE
rknn-toolkit2     x.x.x-cpxx     xxxxxxxxxxxx   1 hours ago    5.89GB
```

5.3 RKNN Python Demo 使用方法

注：请确保板端已经安装 RKNPU2 环境，如果没有，请参考第 3.4 章节中的内容进行安装。

- 映射文件，并运行容器

  请参考第 3.1 章节在本地下载好 RKNN Model Zoo 工程，然后将其映射进容器中，并通过 docker run 命令运行容器。运行后将进入容器的 bash 环境。参考命令如下：

```
# 使用 docker run 命令创建并运行 RKNN Toolkit2 容器
# 并通过附加 -v <host src folder>:<image dst folder> 参数，将本地文件映射进容器中
docker run -t -i --privileged \
    -v /dev/bus/usb:/dev/bus/usb \
    -v /Projects/rknn_model_zoo:/rknn_model_zoo \
    rknn-toolkit2:x.x.x-cpxx \
    /bin/bash
```

- 准备模型

  进入 rknn_model_zoo/examples/yolov5/model 目录，运行 download_model.sh 脚本，该脚本将下载一个可用的 YOLOv5 ONNX 模型，并存放在当前 model 目录下，参考命令如下：

```
# 进入 rknn_model_zoo/examples/yolov5/model 目录
cd rknn_model_zoo/examples/yolov5/model

# 运行 download_model.sh 脚本，下载 yolov5 onnx 模型
# 例如，下载好的 onnx 模型存放路径为 model/yolov5s_relu.onnx
./download_model.sh
```

- 模型转换

  进入 rknn_model_zoo/examples/yolov5/python 目录，运行 convert.py 脚本，该脚本将原始的 ONNX 模型转成 RKNN 模型，参考命令如下：

```
# 进入 rknn_model_zoo/examples/yolov5/python 目录
cd Projects/rknn_model_zoo/examples/yolov5/python

# 运行 convert.py 脚本，将原始的 ONNX 模型转成 RKNN 模型
# 用法: python convert.py model_path [rv1103|rv1103b|rv1106|rv1106b] [i8/fp] [output_path]
python convert.py ../model/yolov5s_relu.onnx rv1106 i8 ../model/yolov5s_relu.rknn
# 注: rv1103、rv1106和rv1103b、rv1106b生成的模型不能共用
```

- 运行 RKNN Python Demo

  进入 rknn_model_zoo/examples/yolov5/python 目录，运行 yolov5.py 脚本，便可通过连板调试的方式在板端运行 YOLOv5 模型，参考命令如下：

```
# 进入 Projects/rknn_model_zoo/examples/yolov5/python 目录
cd Projects/rknn_model_zoo/examples/yolov5/python

# 运行 yolov5.py 脚本，在板端运行 yolov5 模型
# 用法: python yolov5.py --model_path {rknn_model} --target {target_platform}
# 注: 这里以 RV1106 平台为例，如果是其他开发板，则需要修改命令中的平台类型
python yolov5.py --model_path ../model/yolov5s_relu.rknn --target rv1106
```

脚本运行成功后输出信息如下，其中，class 字段代表预测的类别，score 为得分，(xmin, ymin) 是检测框的左上角坐标，(xmax, ymax) 是检测框的右下角坐标。

```
# class @ (xmin, ymin, xmax, ymax) score
person @ (209 243 286 510) 0.880
person @ (479 238 560 527) 0.871
person @ (109 238 231 534) 0.840
bus    @ (91 129 555 464) 0.692
```

6 常见问题

6.1 命令 adb devices 查看不到设备

可以尝试以下方式来解决此问题：

1. 检查连线是否正确、重新插拔数据线、换计算机另一个 USB 端口来连接数据线、更换数据线。
2. 当使用 USB 连接开发板时，请确保本地计算机和 Docker 容器中同时只开启一个 adb server 服务。例如，如果需要在Docker 容器中连接开发板，请在计算机的终端中执行命令 adb kill-server终止本地计算机上的 adb server 服务。
3. 出现以下错误时，表示系统中未安装 adb。需要执行安装命令 sudo apt install adb 安装 adb。

```
command 'adb' not found, but can be installed with:
sudo apt install adb
```

6.2 手动启动 rknn_server 服务

RKNN-Toolkit2 的连板调试功能要求板端已安装 RKNPU2 环境，并且启动 rknn_server 服务。但有些开发板系统没有开机自启动 rknn_server 服务，这导致在运行 RKNN Python Demo 时会遇到如 “E init_runtime: The rknn_server on the concected device is abnormal.” 的报错信息。此时请参考第 3.4.2 章 “检查 RKNPU2 环境是否安装” 中启动 rknn_server 服务的相关内容手动启动 rknn_server 服务。

6.3 内存不足导致模型加载失败

RV1103/RV1103B/RV1106/RV1106B 开发板上内存较小，如果此时开发板上已经运行 rkipc 进程，则可能导致模型加载失败，遇到 “E init_runtime: Exception: RKNN init failed. error code: RKNN_ERR_MODEL_INVALID” 报错信息。此时需要进入板端，手动结束掉 rkipc 进程，参考命令如下：

```
# 进入板端
adb shell

# 查看当前运行的进程，找到 rkipc 进程对应的进程ID（PID）
top

# 结束 rkipc 进程
kill -9 PID
```

7 参考文档

- 有关RKNN-Toolkit2和RKNPU2更详细的用法和接口说明，请参考《Rockchip_RKNPU_User_Guide_RKNN_SDK_CN.pdf》手册。